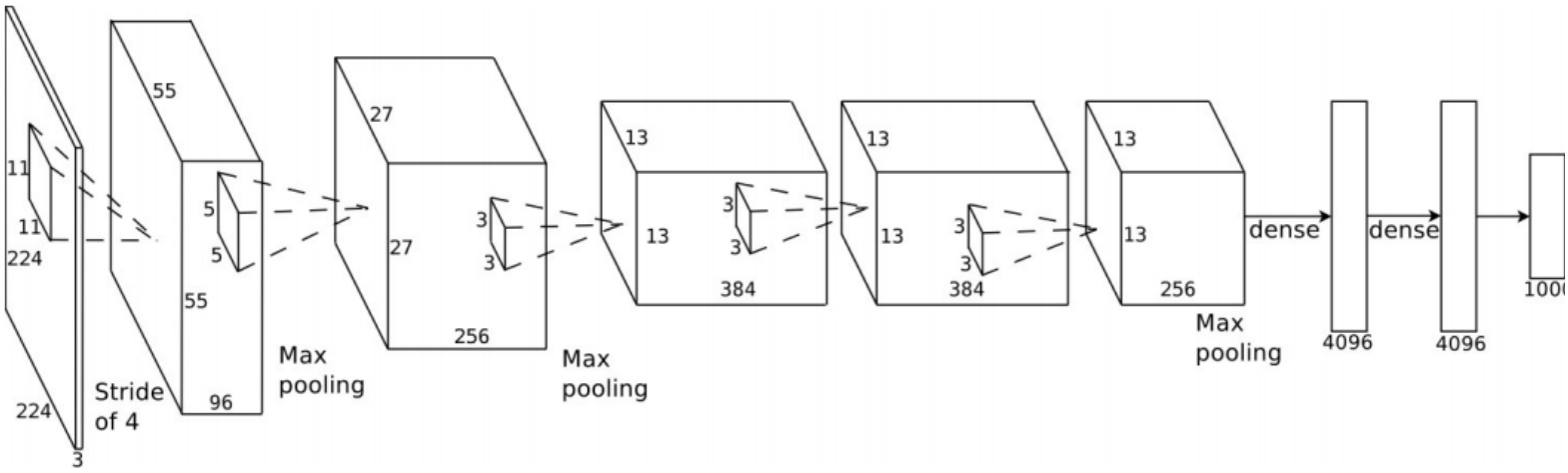
Parallelizing Convolutional Neural Networks

Alex Krizhevsky
Google

June 23, 2014

Convolutional neural networks



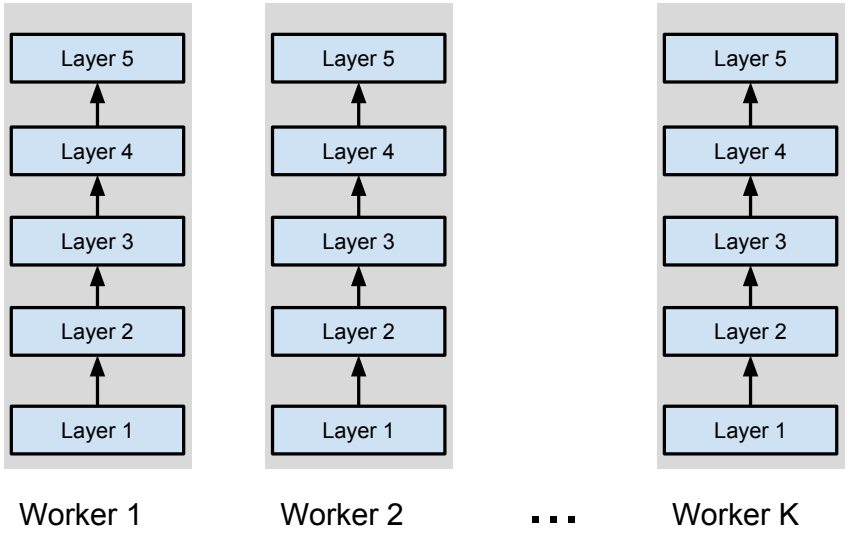
Motivations

- Training a neural net is the inner loop in research and architecture exploration.
- Faster training broadens the space of things you're willing to try.
- Big datasets take a lot of time to consume.

The basic algorithm

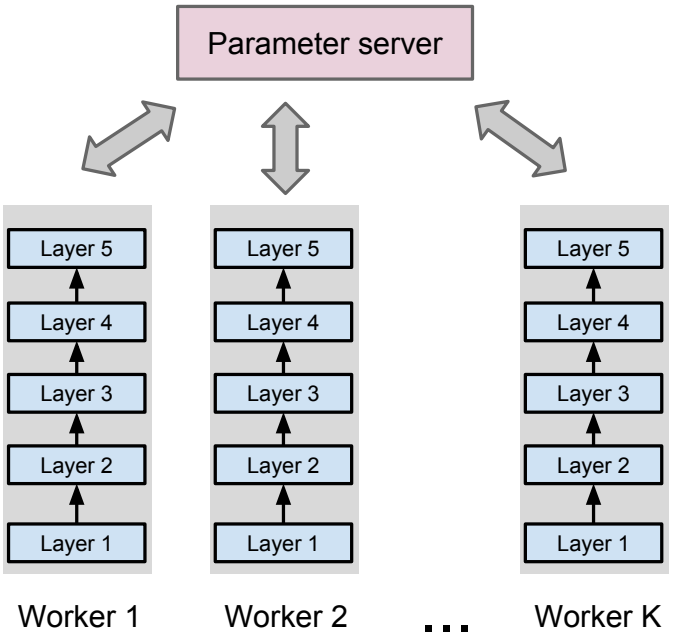
```
for training batch i in {1 … N}
  w += -E[dE/dw|i]
```

Data parallelism



- Workers train the same model on different data examples
- Share weights or gradients
- Batch size increases with number of workers

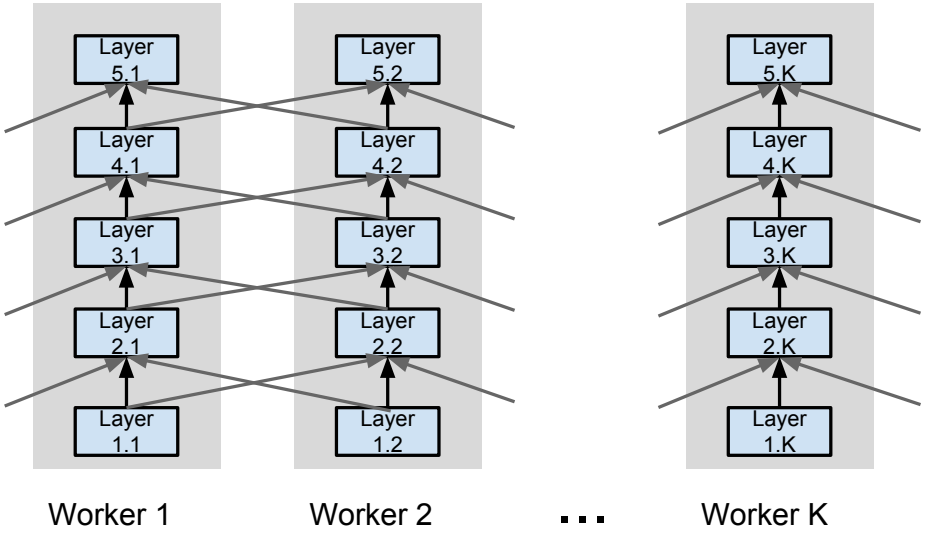
Asynchronous data parallelism



- Workers asynchronously write parameter updates to shared memory or dedicated server
- Efficient when gradients are very sparse

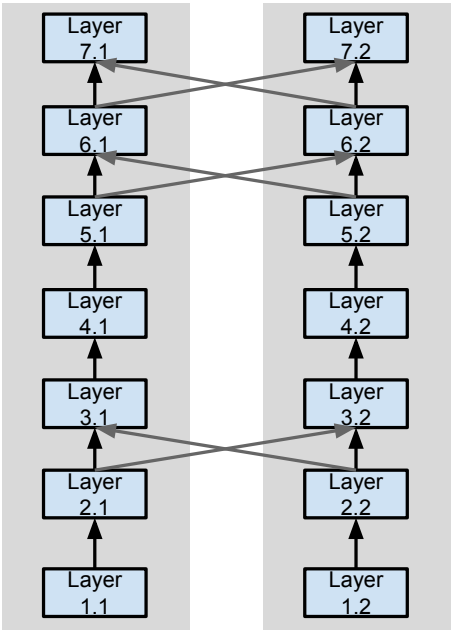
Citations: Hogwild, DistBelief

Model parallelism

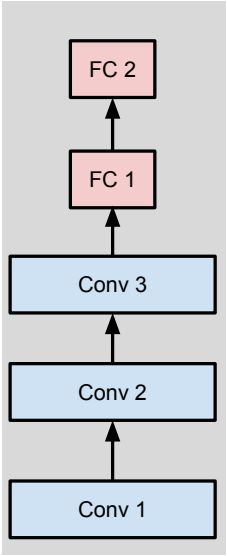


- Workers train different parts of the same model on the same data examples
- Workers share neuron activations

Old ImageNet network did this



Applied to convolutional nets



- Model parallelism with “multi-tower” models
  - Lots of low-level filters
  - Convolutional layers have many neurons, so exchanging them is expensive
- Data parallelism with big batch sizes or async SGD
  - Generally worse convergence
  - Expensive gradient communication

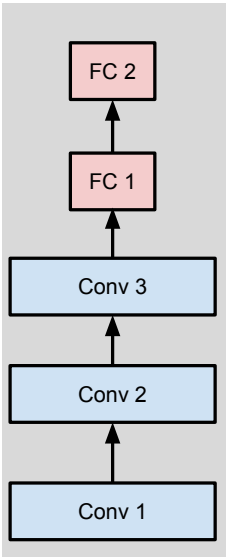
Data parallelism

- Workers exchange messages of size proportional to **number of weights**
- Efficient when the amount of **computation per weight** is high

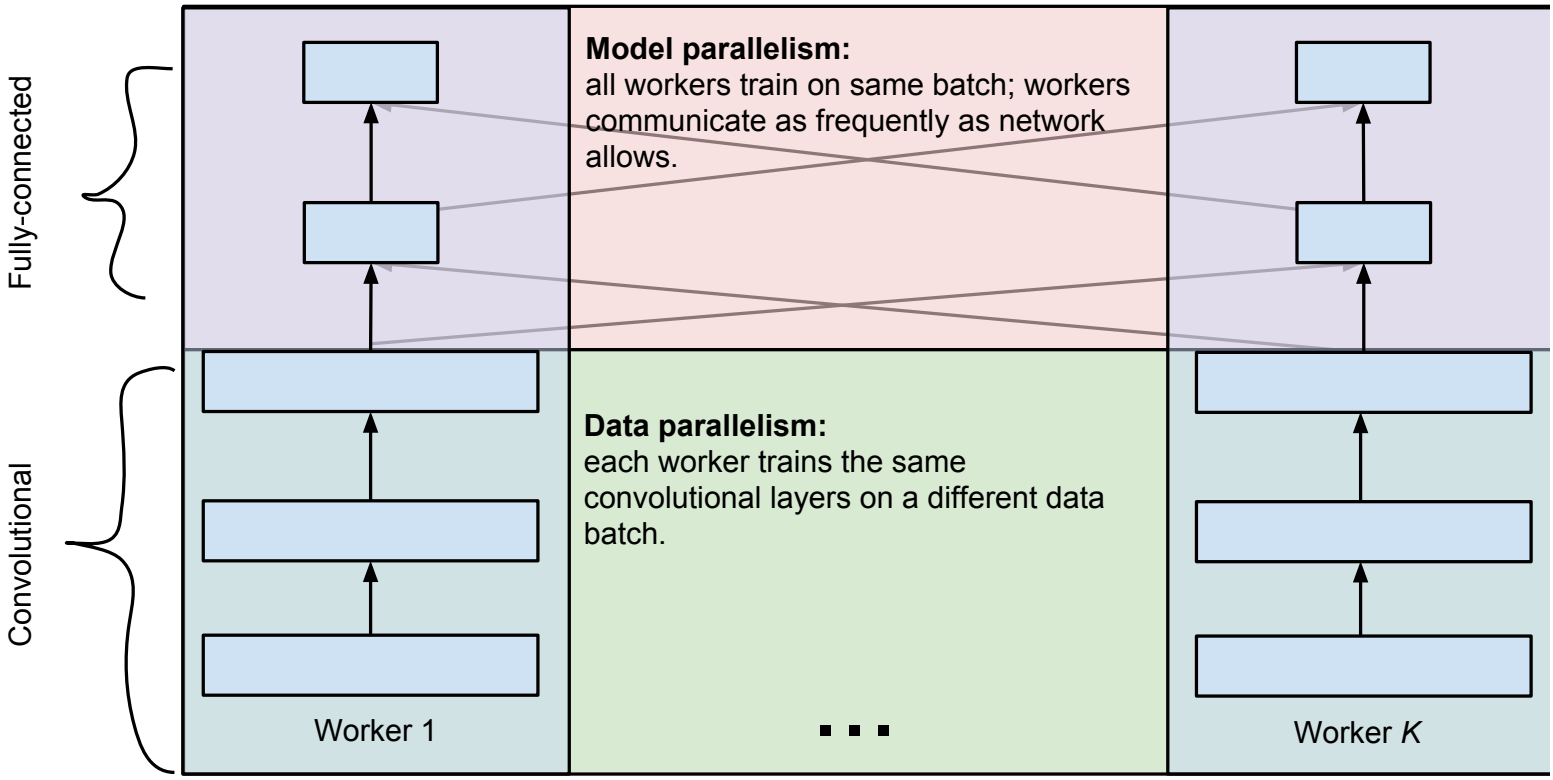
Model parallelism

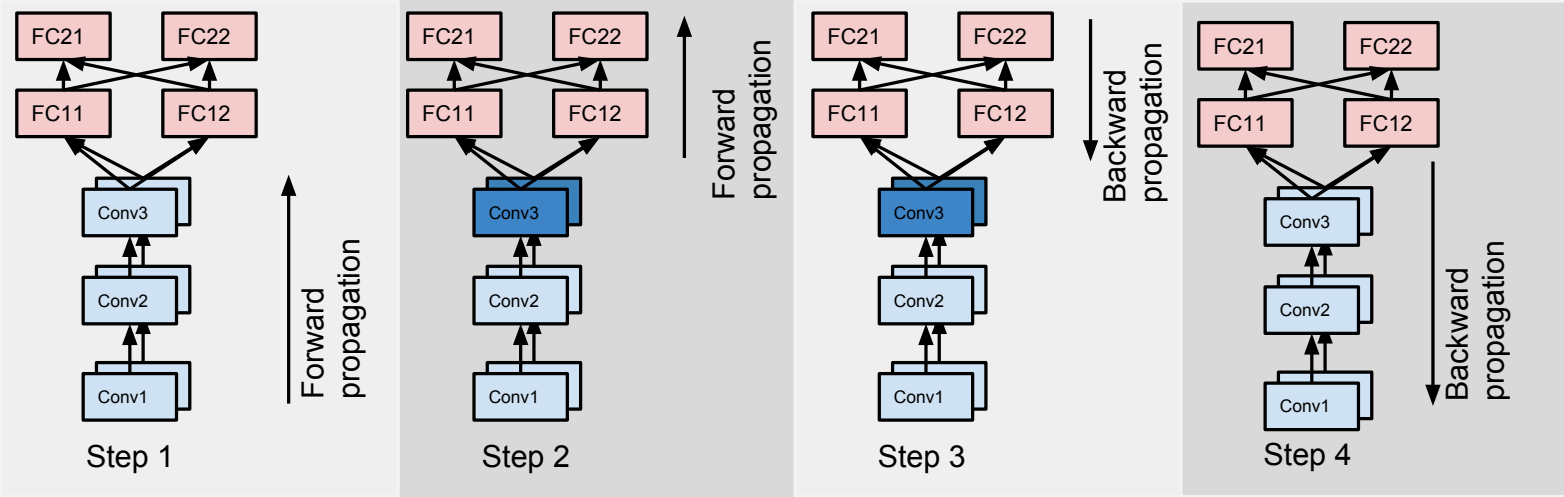
- Workers exchange messages of size proportional to **number of neurons**
- Efficient when the amount of **computation per neuron** is high

Convolutional neural nets

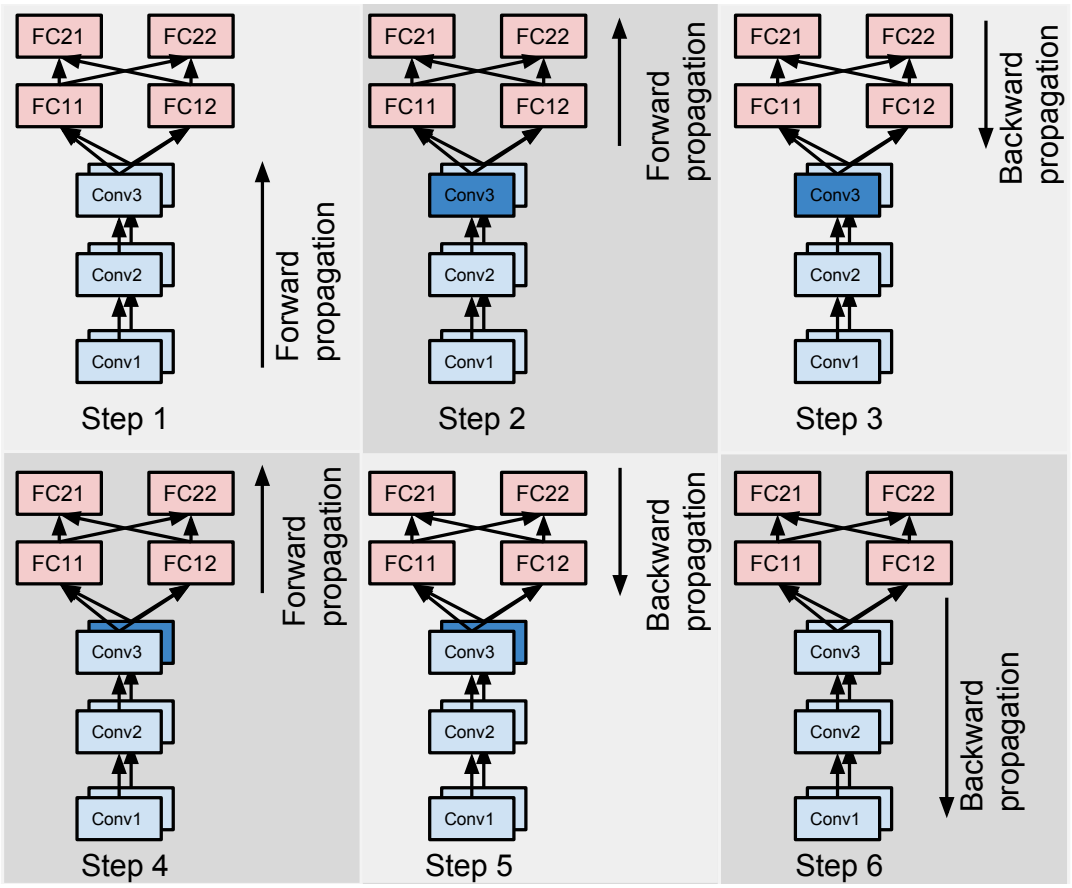


	Convolutional	Fully-connected
Computation	★★★★	★
Neurons	★★★★	★
Weights	★	★★★★


Fully-connected
Convolutional
Model parallelism:
all workers train on same batch; workers communicate as frequently as network allows.
Data parallelism:
each worker trains the same convolutional layers on a different data batch.
Worker 1
Worker K



Each worker sends **Conv3** activations to other worker and they do a forward pass on the big batch of activations


FC21
FC22
FC11
FC12
Conv3
Conv2
Conv1
Forward propagation
Step 1
Forward propagation
Step 2
Backward propagation
Step 3
Forward propagation
Step 4
Backward propagation
Step 5
Backward propagation
Step 6

Properties

- The one big communication is overlapped with computation
- Can update ~90% of the weights very frequently -- as though you have a small batch size
- Lots of freedom in choosing fully-connected connectivity for efficiency

Performance on 2012 network

GPUs	Batch size	Top-1 error	Time	Speedup
1	(128, 128)	42.33%	98.05h	1x
2	(256, 256)	42.63%	50.24h	1.95x
2	(256, 128)	42.27%	50.90h	1.93x
4	(512, 512)	42.59%	26.20h	3.74x
4	(512, 128)	42.44%	26.78h	3.66x
8	(1024, 1024)	43.28%	15.68h	6.25x
8	(1024, 128)	42.86%	15.91h	6.16x

Code soon

- An update to cuda-convnet
  - It's called cuda-convnet2
- Faster training on modern Nvidia GPUs (GeForce Titan, K20, K40)
- Multi-GPU support implementing the forms of parallelism discussed here.